



Clase 4. Python

Controladores de Flujo 1

***RECUERDA PONER A GRABAR LA
CLASE***





OBJETIVOS DE LA CLASE

- Conceptualizar el flujo y diagrama de flujo
- Reconocer sus funcionalidades
- Conceptualizar sentencias de control
- Utilizar sentencia de control if

CRONOGRAMA DEL CURSO

Clase 3



Operadores y expresiones



OPERADORES RELACIONALES



OPERADORES LÓGICOS



EXPRESIONES ANIDADAS

Clase 4



Controladores de flujo 1



MAYORÍA DE EDAD



MARVEL VS CAPCOM

Clase 5



Controladores de flujo 2



NÚMEROS

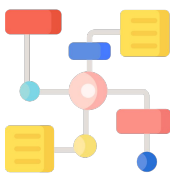


EJERCICIOS



CONTROL DE FLUJO

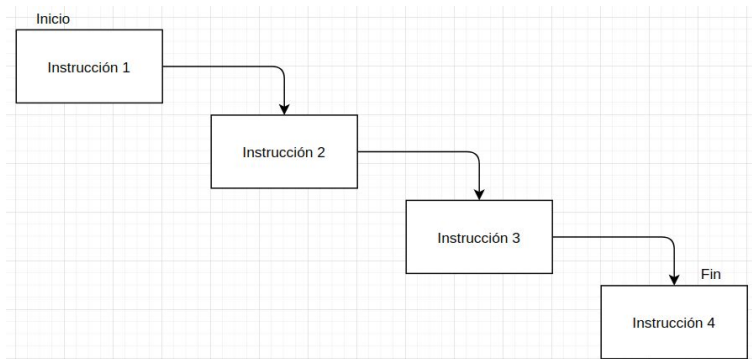
FLUJO

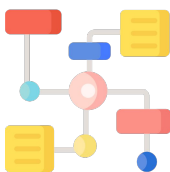


¿Qué es el Flujo?

Veremos cómo controlar el flujo con python, pero antes de eso, ¿Qué es el flujo?

El flujo **es una forma de entender la sucesión de las instrucciones de un programa**, estas instrucciones se ejecutan una después de otras de forma ordenada y suelen tener el objetivo final de manipular información

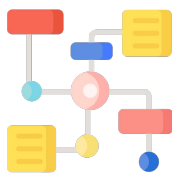




¿Qué es el Flujo?

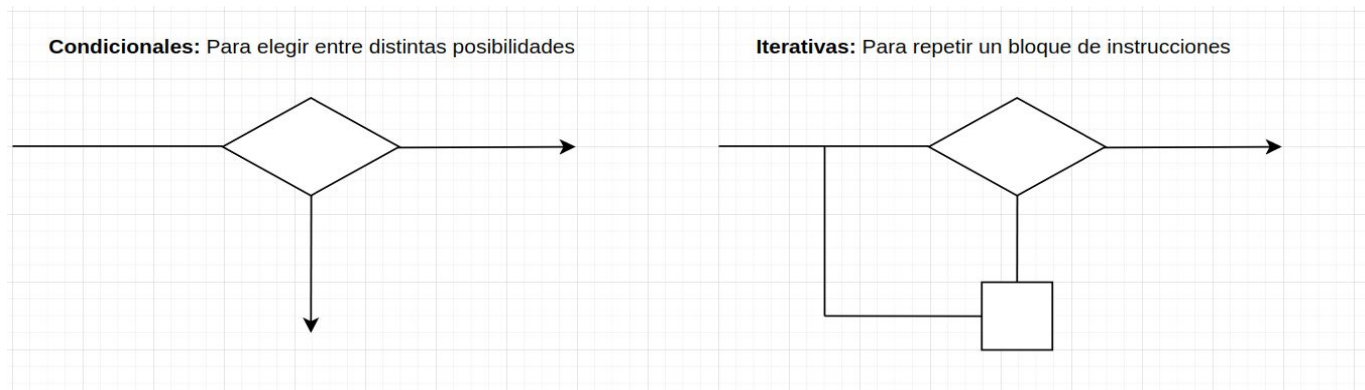
Sin embargo, para manipular datos no es suficiente con realizar cálculos o resolver expresiones, **necesitamos que de alguna forma nuestro programa pueda elegir, que sepa cómo actuar en función de determinadas situaciones**, o incluso repetir una tarea si es necesario.

Para estas situaciones, existen las **sentencias de control de flujo**.



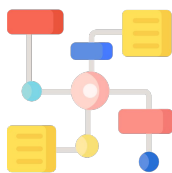
Sentencias de control

Se dividen en dos tipos, las de **control condicional** y las de **control iterativo**. (A las siguientes imágenes se le denomina **diagrama de flujo**)



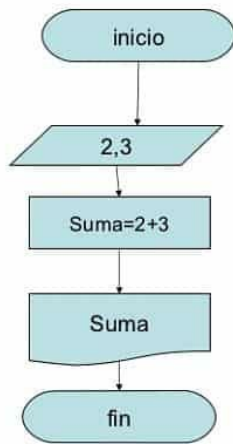
Nos centraremos en las **sentencias de control condicional**.

CODER HOUSE



Diagramas de Flujo

Expresan nuestros algoritmos en forma de diagrama mediante una representación gráfica basada en figuras geométricas que varían según la estructura de código.



Punto de inicio del programa

Entrada de datos 2,3

Proceso

Salida

Fin

Una app recomendada que se suele utilizar es

[Diagrams](#) 😊

CONDICIONAL



Condicional

En la vida diaria, actuamos de acuerdo a la evaluación de condiciones, de manera mucho más frecuente de lo que en realidad creemos: Si el semáforo está en verde, cruzar la calle. Si no, esperar a que el semáforo se ponga en verde.

A veces, también evaluamos más de una condición para ejecutar una determinada acción: Si llega la factura de la luz y tengo dinero, pagar la factura.



CODER HOUSE



Condicional

Las sentencias de control condicionales, son aquellas que nos permiten evaluar si una o más condiciones se cumplen, para decir qué acción vamos a ejecutar. La evaluación de condiciones, sólo puede arrojar 1 de 2 resultados: **True** o **False** (verdadero o falso).



Condicional

Para describir la evaluación a realizar sobre una condición, se utilizan los **operadores relacionales** (`==`, `!=`, `>`, `<`, etc). Y, para evaluar más de una condición simultáneamente se utilizan los **operadores lógicos** (`not`, `and`, `or`).

Las sentencias de control de flujo condicionales se definen mediante el uso de tres palabras claves reservadas:

👉 Pasemos a ver cada una

if (sí)

elif (sino, sí)

else (sino)



Sentencia If

Dentro de las sentencias condicionales el **if (si)** posiblemente sea la más famosa y utilizada en la programación, esto debido a que nos permite controlar el flujo del programa y dividir la ejecución en diferentes caminos.



Sentencia If

Al utilizar esta palabra reservada **if** le estamos indicando a Python que queremos ejecutar una porción de código, o bloque de código, sólo si se cumple una determinada condición, es decir, si el resultado es **True**.



Sentencia If

Veamos el siguiente ejemplo:

```
>>> edad = 24
>>> if edad >= 18:
    print("Es un adulto.")
```

Primero definimos una variable **edad** y le asignamos un valor entero **24**. Después, a través del condicional, le decimos que queremos imprimir **“Es un adulto”** en pantalla, sólo si se cumple la condición de que **edad** sea mayor o igual a 18.

```
>>> if True:
    print("Se cumple la condición.")
```

CODER HOUSE



Indentación

En python definimos un bloque de código después de una sentencia de control con dos puntitos **:** y todo el bloque que se ejecutará debe de estar **indentado**. Esto le indicará a python que el código **indentado** está dentro del bloque de código, y solamente se ejecutará si se cumple la condición.

```
>>> if edad >= 18:  
    print("Es un adulto.")  
    print("Es un adulto.")
```



If

Mientras el **if** siempre reciba una expresión lógica o un valor lógico **True** siempre se ejecutará.

```
>>> if False:
```

*¿Y si le pasamos **False**?*

```
    print("Se cumple la condición.")
```

```
    print("Otro print a mostrar.")
```

CODER HOUSE



If

¿Se acuerdan de lo que podíamos hacer para modificar el valor de **False**?

El operador lógico **not** niega los valores, esto también afecta al True y False, haciendo que sean lo opuesto:

```
>>> if not False:
```

```
    print("Ahora si se cumple la condición.")
```

CODER HOUSE



Múltiples IF

Probemos ahora una expresión que devuelva True o False en lugar de un tipo lógico y a la vez intentemos crear más **if** (podemos encadenar más ifs!).

```
>>> if a == 4:  
    print("a vale 4")  
    if a == 5:  
        print("a vale 5")
```

Falsa!

Verdadera!!



¿If dentro de if?

Podemos definir múltiples **if**'s dentro de otros **if**'s siempre y cuando respetemos los niveles de indentación.

```
>>> a = 5:
>>> b = 10:
>>> if a == 2:
    print("a vale ", a)
    if b == 10:
        print("y b vale ", b)
```



//f

Pero, ¿**Podemos hacer ambas comprobaciones a la vez?** ¿No sería más fácil?

¡Si! Por eso existe el operador lógico **and**

```
>>> a = 5:  
>>> b = 10:  
>>> if a == 2 and b == 10:  
    print("a vale ", a, "y b vale ", b)
```

ELSE



Sentencia Else

Dentro de las sentencias condicionales el **else** (sino) es una especie de “hermano” de **if** el cual se puede encadenar al final de un bloque de código **if** para comprobar los casos contrarios, es decir los **False**



Sentencia Else

Al utilizar esta palabra reservada **else** le estamos indicando a Python que queremos ejecutar una porción de código, o bloque de código, sólo si no se cumple ninguna de las condiciones antes dichas, es decir, si el resultado es **False** siempre.



Sentencia Else



Veamos el siguiente ejemplo:

```
>>> numero = 24
>>> if numero > 36:
    print("El número es grande.")
else:
    print("El número es chico.")
```

1. Primero definimos una variable numero y le asignamos un **valor entero 24**.
2. Después, a través del condicional, le preguntamos si el número es **mayor a 36**, si es así, queremos **imprimir "El número es más grande"** en pantalla, **de lo contrario** queremos que **imprima "El número es más chico"**.



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

ELIF



Sentencia Elif

La última sentencia condicional que podemos encontrar es el **elif** (sino, si), también podríamos decir que es un hermano de **if** ya que se utiliza en continuación al **if** para poder encadenar muchísimas más comprobaciones.



Sentencia Elif

Al utilizar esta palabra reservada **elif** le estamos indicando a Python que queremos ejecutar una porción de código, o bloque de código, sólo si la condición anterior no se cumple, es decir, si el resultado del **if** o algún **elif** fue **False**.



Sentencia Elif



Veamos el siguiente ejemplo:

```
>>> edad = 24
>>> if edad >= 36:
    print("Es un adulto.")
elif edad == 24:
    print("La edad es 24")
else:
    print("No sabemos la
edad")
```

1. Primero definimos una **variable edad** y le asignamos un **valor entero 24**.
2. Después, a través del condicional, le decimos que queremos **imprimir "Es un adulto"** en pantalla, **sólo si** se cumple la condición de que **edad sea mayor o igual a 36**.
3. Si **edad es igual a 24**, **imprimimos "La edad es 24"**. Sino, **imprimimos que no sabemos la edad**.

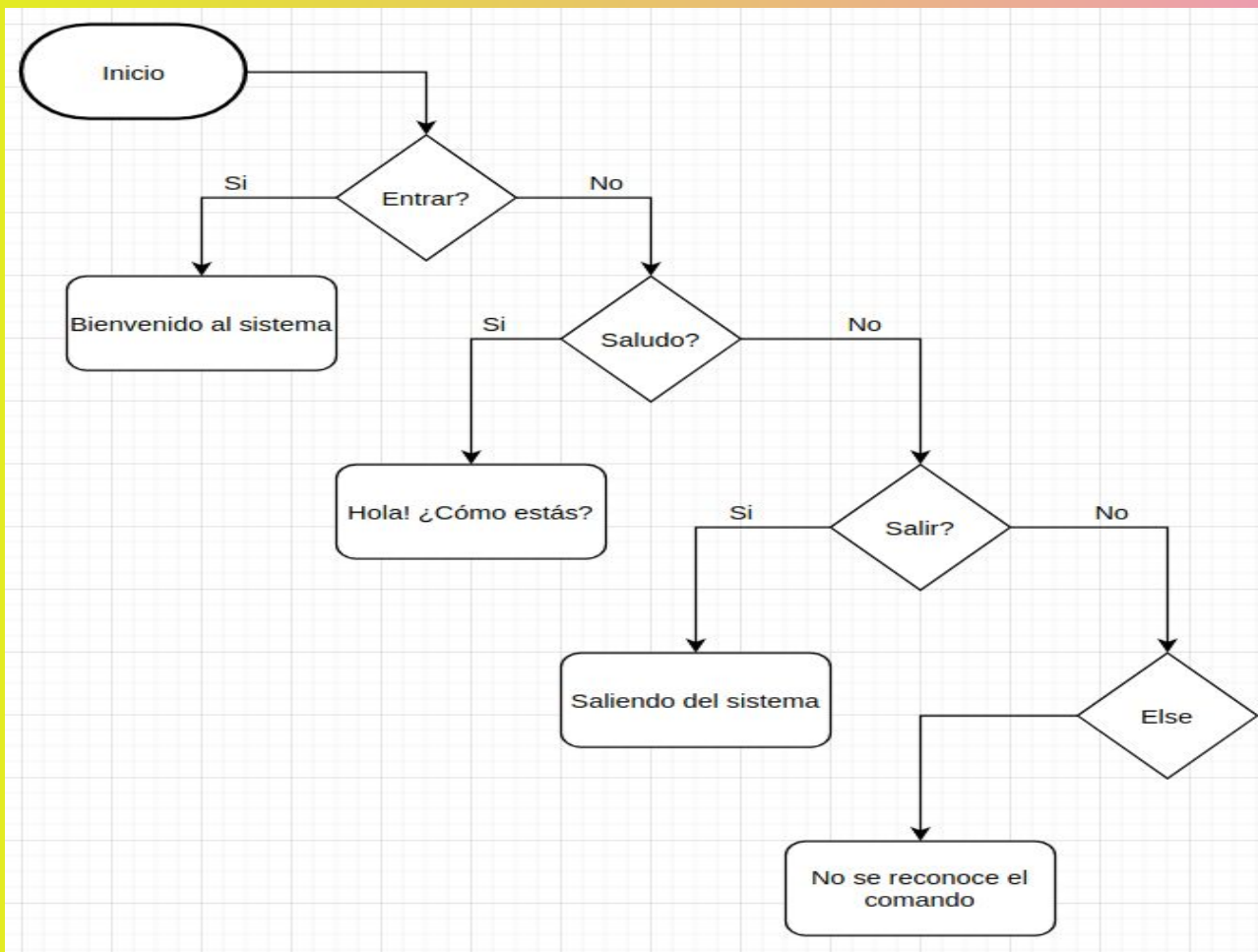


¿Para qué sirve la sentencia Elif?



```
>>> comando = "SALIR"
>>> if comando == "ENTRAR":
    print("Bienvenido al sistema.")
elif comando == "SALUDO":
    print("Hola! ¿Cómo estás?")
elif comando == "SALIR":
    print("Saliendo del sistema.")
else:
    print("No se reconoce el comando.")
```

Básicamente nos sirve para
poder darle múltiples
opciones al programa





Cuando se tiene varios **if**'s se ven las múltiples condiciones y si todo está bien, nos mostrará el resultado de cada **if**.

Sin embargo, en el caso de múltiples **elif**, comprueba las condiciones de arriba a abajo hasta que se cumpla una de ellas, y de ser así, las demás no se comprueban.



Sentencias

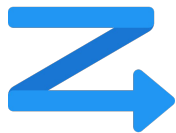
```
>>> nota = 9
>>> if nota >= 9:
    print("Sobresaliente.")
if nota >= 7:
    print("Muy bien")
if nota >= 6:
    print("Bien")
if nota >= 4:
    print("Regular")
else:
    print("Insuficiente")
```

Esto nos devolverá:

**“Sobresaliente”, “Muy bien”, “Bien”,
“Regular”**

¿Por qué nos muestra 4 prints? Porque el **if** recorre cada comprobación, si se cumple, ingresa y ejecuta el código, y luego pasa a la siguiente comprobación.

CODER HOUSE



Sentencias

```
>>> nota = 9
>>> if nota >= 9:
    print("Sobresaliente.")
elif nota >= 7:
    print("Muy bien")
elif nota >= 6:
    print("Bien")
elif nota >= 4:
    print("Regular")
else:
    print("Insuficiente")
```

Esto nos devolverá: **"Sobresaliente"**

¿Por qué nos muestra 1 print? Por qué el **elif** recorre cada comprobación, si se cumple una, ingresa, realiza lo escrito en el bloque de código, y sale del programa, es decir, no ejecuta ningún código más.

CODER HOUSE

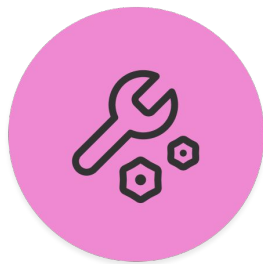


¿Podemos arreglar los ifs?



```
>>> nota = 9
>>> if nota >= 9:
    print("Sobresaliente.")
    if nota >= 7 and nota < 9:
        print("Muy bien")
    if nota >= 6 and nota < 7:
        print("Bien")
    if nota >= 4 and nota < 6:
        print("Regular")
    else:
        print("Insuficiente")
```

Limitando la comprobación podemos imitar al elif, pero **no nos conviene** debido a que el elif es muchísimo más fácil de utilizar y no debemos limitar nada.



Mayoría de edad

Escribir un programa

Tiempo estimado: 5 minutos



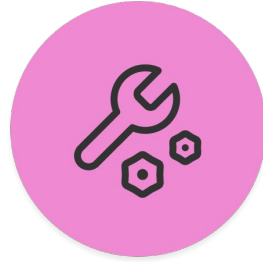
Mayoría de Edad

Tiempo estimado: 5 minutos

Escribir un programa que le pregunte al usuario su edad y muestre por pantalla si es mayor de edad o no.

Nota:

Para preguntarle al usuario, recuerda usar input



Marvel vs CapCom

Tiempo estimado: 20 minutos

Marvel vs CapCom

Desafío
generico



Tiempo estimado: 20 minutos

Un curso se ha dividido en dos grupos: **A** y **B** de acuerdo al nombre y a una preferencia (**Marvel o Capcom**). El **grupo A** está formado por fans de Marvel con un nombre anterior a la M y los fans de Capcom con un nombre posterior a la N y el **grupo B** por el resto. Escribir un programa que pregunte al usuario su nombre y preferencia, y muestre por pantalla el grupo que le corresponde.

Ej:

¿Cómo te llamas? Alan

¿Cuál es tu preferencia (M o C)? C

Tu grupo es B

Para preguntarle al usuario, recuerda usar **input**

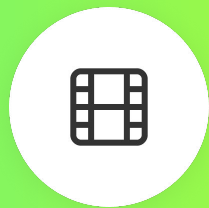


EJEMPLO SUBIDO AL DRIVE: [Desafio](#)

CODER HOUSE

¿PREGUNTAS?





***¿QUIERES SABER MÁS? TE DEJAMOS
MATERIAL AMPLIADO DE LA CLASE***

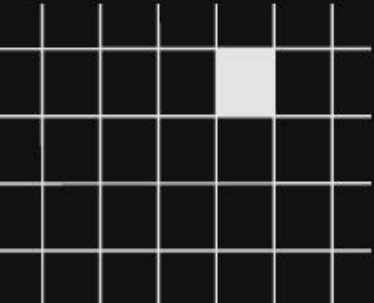


- Artículo: [Pseudocódigo y Diagramas de flujo](#)
- Artículo: [Funciones Listas](#)
- Artículo: [Tipo Tuplas](#)



¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Flujo
 - Sentencia de control
 - Diagrama de Flujo
 - Sentencias
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE